Reminding

- 11/10(Thu) HW2 Announcement
  - TAs will announce the HW2 requirement today

- 11/24(Thu) Final Project Announcement [11/17->11/24]
- 12/8(Thu) HW2 Deadline & HW3 Announcement

  Submit your research topic (for group presentation)

Recurrent Neural Network

顏安孜

azyen@cs.nycu.edu.tw

Some slides are selected from the course material of Machine Learning And Having It Deep And Structured by Prof. Hung-Yi Lee and Applied Deep Learning by Prof. Yun-Nung Chen

Traditional Language models

- N-grams

Bi-grams

$$P(w_2|w_1) = \frac{count(w_1, w_2)}{count(w_1)}$$

Tri-grams

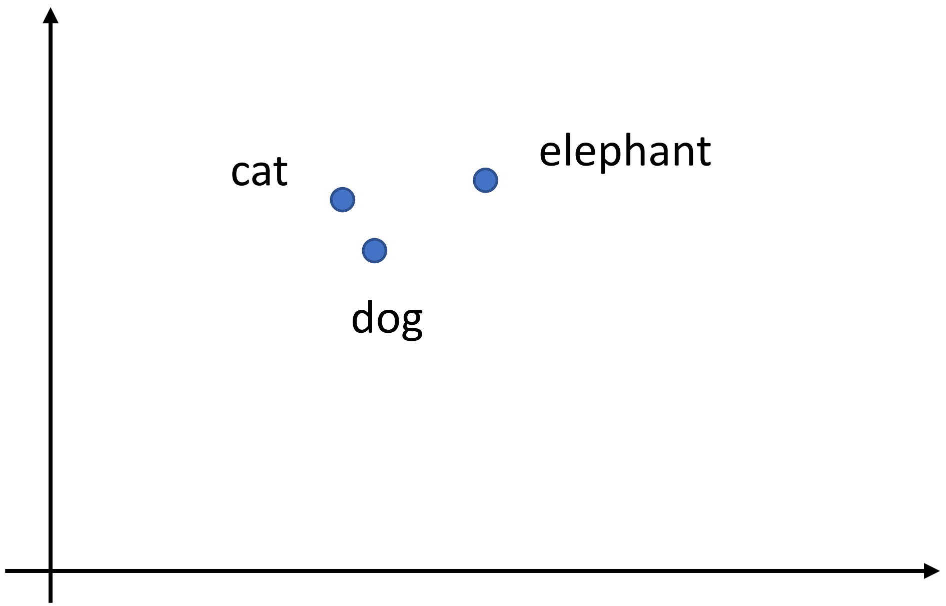
$$P(w_3|w_1, w_2) = \frac{count(w_1, w_2, w_3)}{count(w_1, w_2)}$$

$$P(w_1, w_2, w_3) = P(w_1) \times P(w_2|w_1) \times P(w_3|w_2)$$

- ➔ Large N-grams to capture dependencies between distant words
- ➔ Need a lot of space and RAM

Neural Language Modeling

- The input layer (or hidden layer) of the related words are close



- If P(jump | dog) is large, P(jump | cat) increase accordingly (even there is not "... cat jump ..." in the data)
- Smoothing is automatically done.

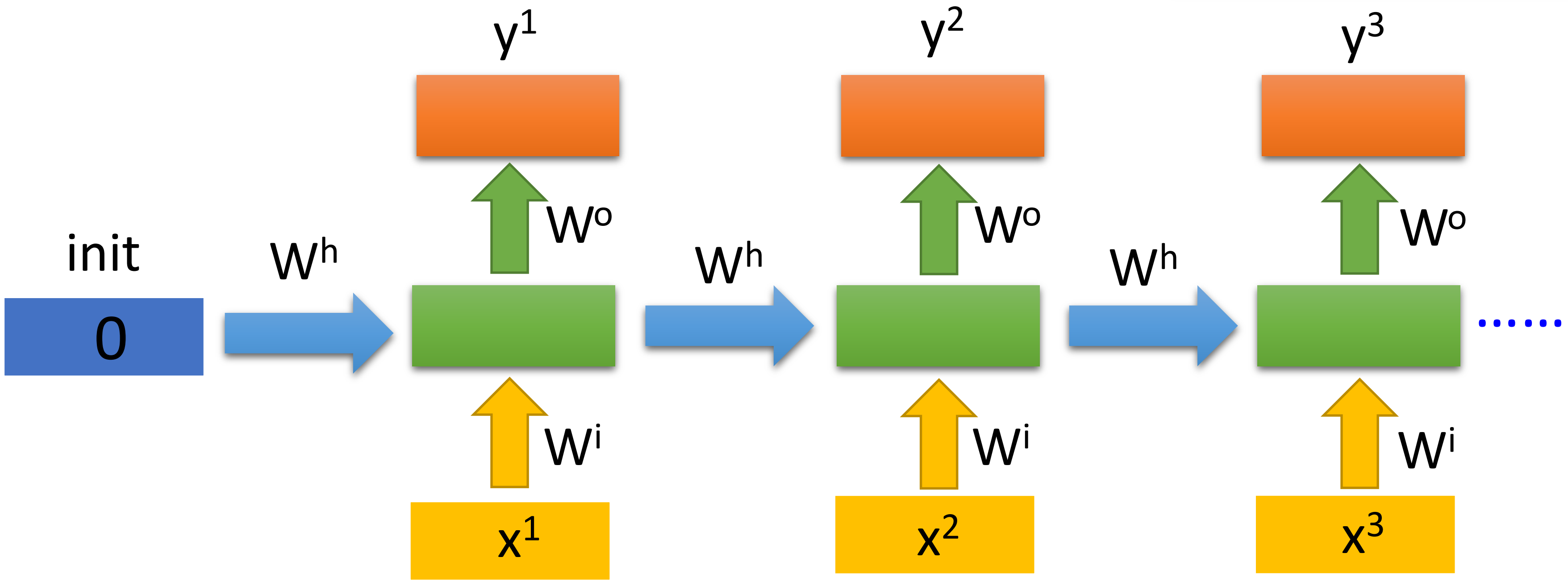
Recurrent Neural Network

- Idea: condition the neural network on all previous words and tie the weights at each time step
- Assumption: temporal information matters

Recurrent Neural Network

Input data: x^1 x^2 x^3 …… x^N

Input of RNN is one utterance



The same network is used again and again.

Output y^i depends on x^1, x^2, …… x^i

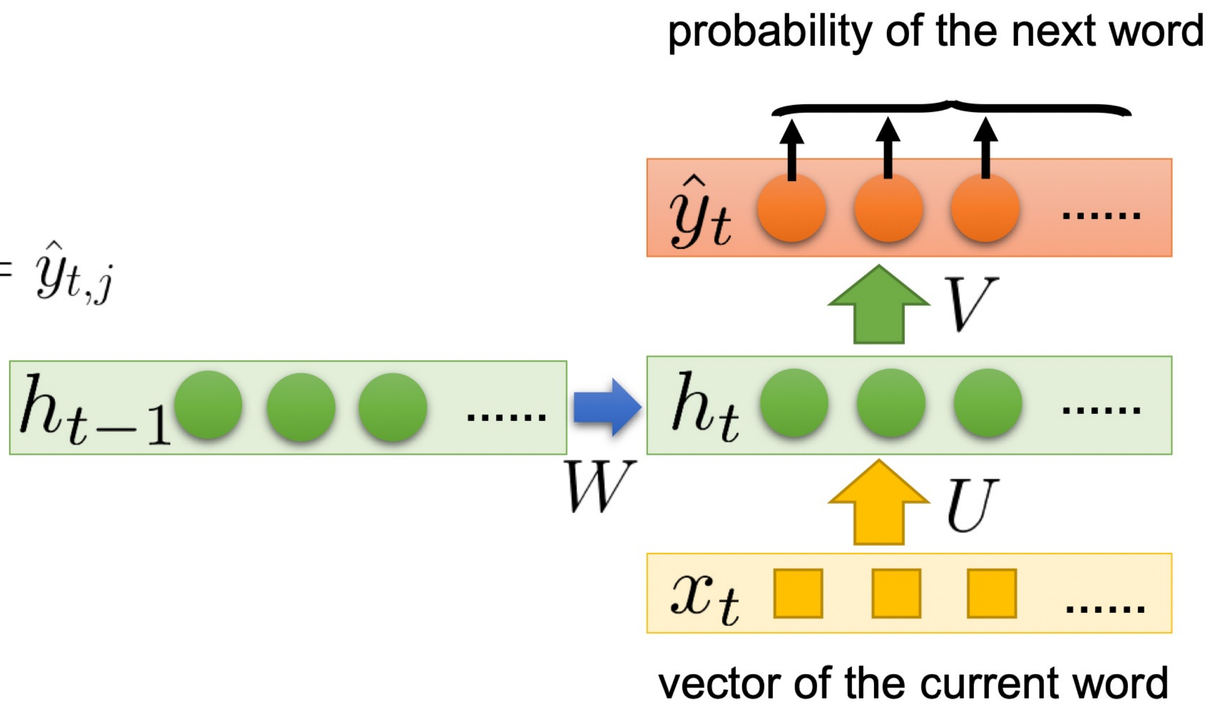
RNNLM Formulation

- At each time step,

$$h_t = \sigma(W h_{t-1} + U x_t)$$

$$\hat{y}_t = \mathrm{softmax}(V h_t)$$

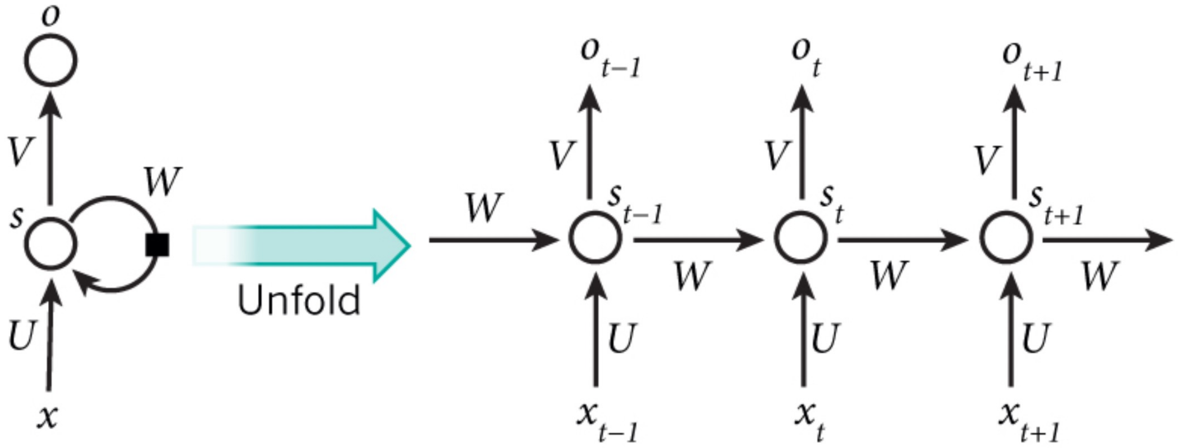
$$P(x_{t+1} = w_j \mid x_1, \cdots, x_t) = \hat{y}_{t,j}$$



Recurrent Neural Network Definition

$$s_t = \sigma(W s_{t-1} + U x_t)$$
$$o_t = \mathrm{softmax}(V s_t)$$

$\sigma(\cdot)$: tanh, ReLU

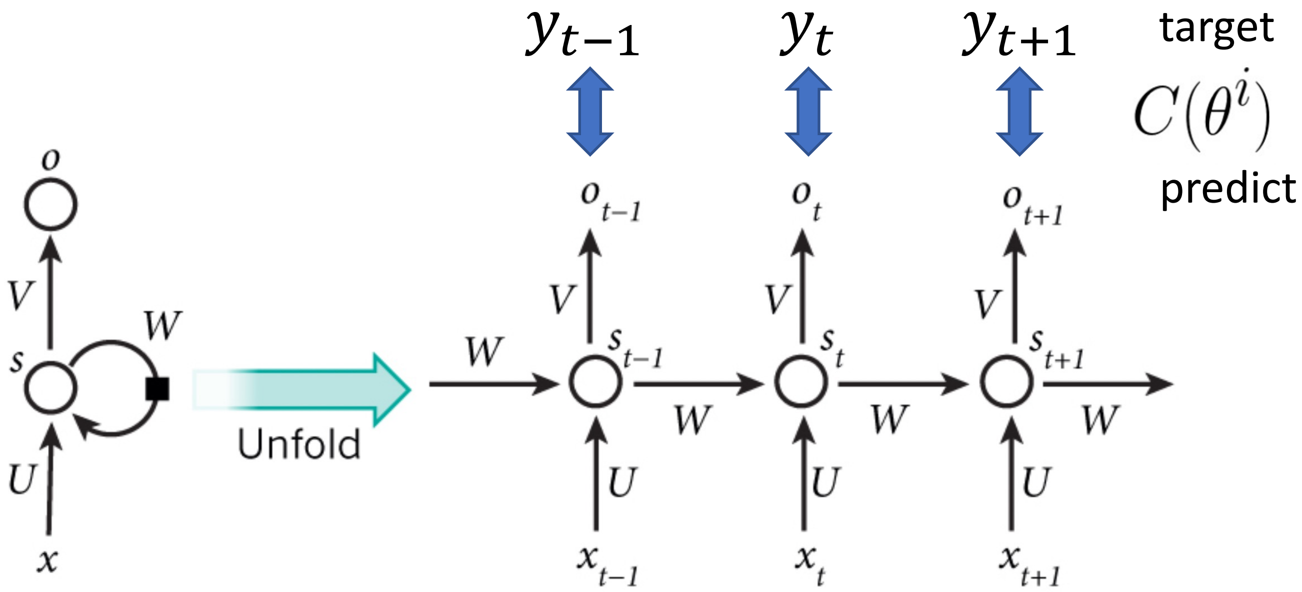


- Source of image: https://dennybritz.com/posts/wildml/recurrent-neural-networks-tutorial-part-1/

Model Training

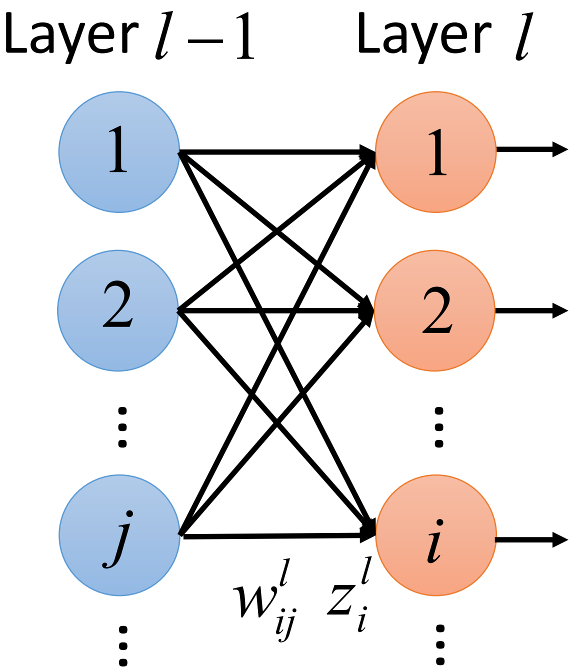
- All model parameters $\theta = \{U, V, W\}$ can be updated by

$$\theta^{i+1} \leftarrow \theta^{i} - \eta \nabla_{\theta} C(\theta^{i})$$



Backpropagation

$$\frac{\partial C_x}{\partial w_{ij}^l} = \boxed{\frac{\partial z_i^l}{\partial w_{ij}^l}}\boxed{\frac{\partial C_x}{\partial z_i^l}}$$



$$\begin{cases} a_j^{l-1} & l > 1 \\ x_j & l = 1 \end{cases}$$

Error signal

$$\delta_i^l$$

Forward Pass

$$z^1 = W^1 x + b^1$$

$$a^1 = \sigma\left(z^1\right)$$

$$\cdots\cdots$$

$$z^{l-1} = W^{l-1} a^{l-2} + b^{l-1}$$

$$a^{l-1} = \sigma\left(z^{l-1}\right)$$

Backward Pass

$$\delta^L = \sigma'\left(z^L\right) \bullet \nabla C_x\left(y\right)$$

$$\delta^{L-1} = \sigma'\left(z^{L-1}\right) \bullet \left(W^L\right)^T \delta^L$$

$$\cdots\cdots$$

$$\delta^l = \sigma'\left(z^l\right) \bullet \left(W^{l+1}\right)^T \delta^{l+1}$$

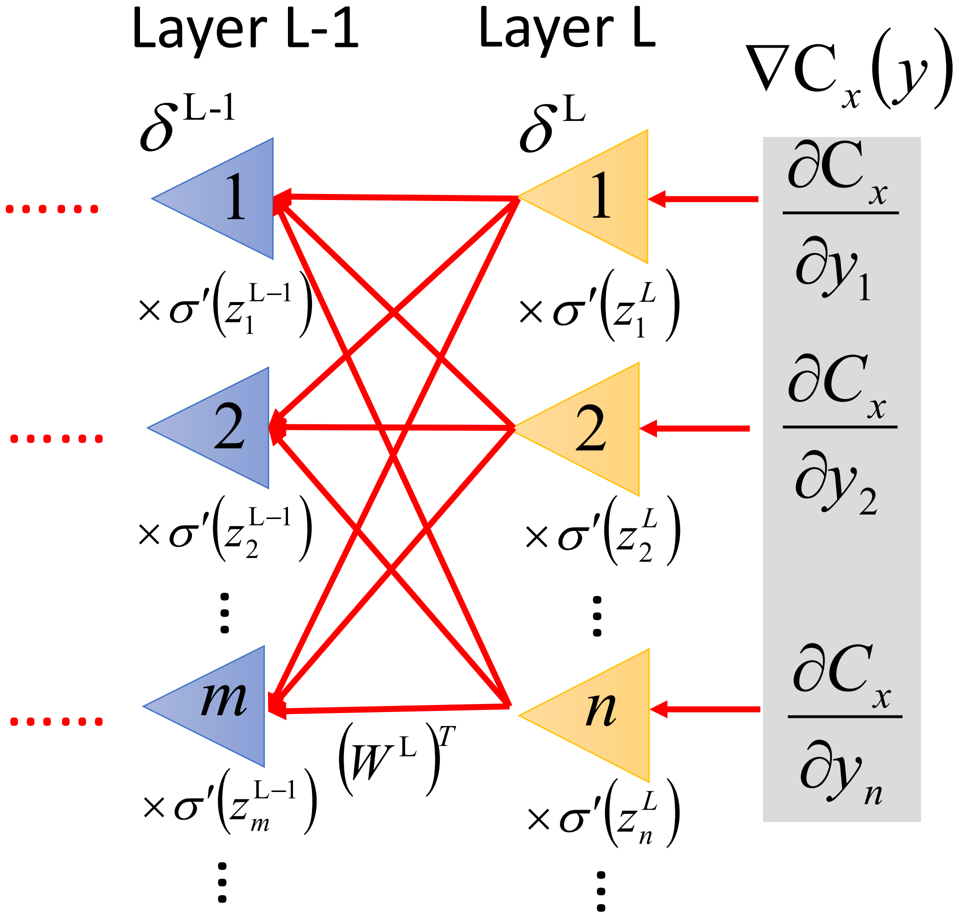
$$\cdots\cdots$$

Backpropagation

$$\frac{\partial C_x}{\partial w_{ij}^l} = \boxed{\frac{\partial z_i^l}{\partial w_{ij}^l}} \boxed{\frac{\partial C_x}{\partial z_i^l}}$$

Error signal

$$\boxed{\delta_i^l}$$



Backward Pass

$$\delta^L = \sigma'(z^L) \bullet \nabla C_x(y)$$

$$\delta^{L-1} = \sigma'(z^{L-1}) \bullet (W^L)^T \delta^L$$

$$\cdots\cdots$$

$$\delta^l = \sigma'(z^l) \bullet (W^{l+1})^T \delta^{l+1}$$

$$\cdots\cdots$$

Backpropagation through Time (BPTT)

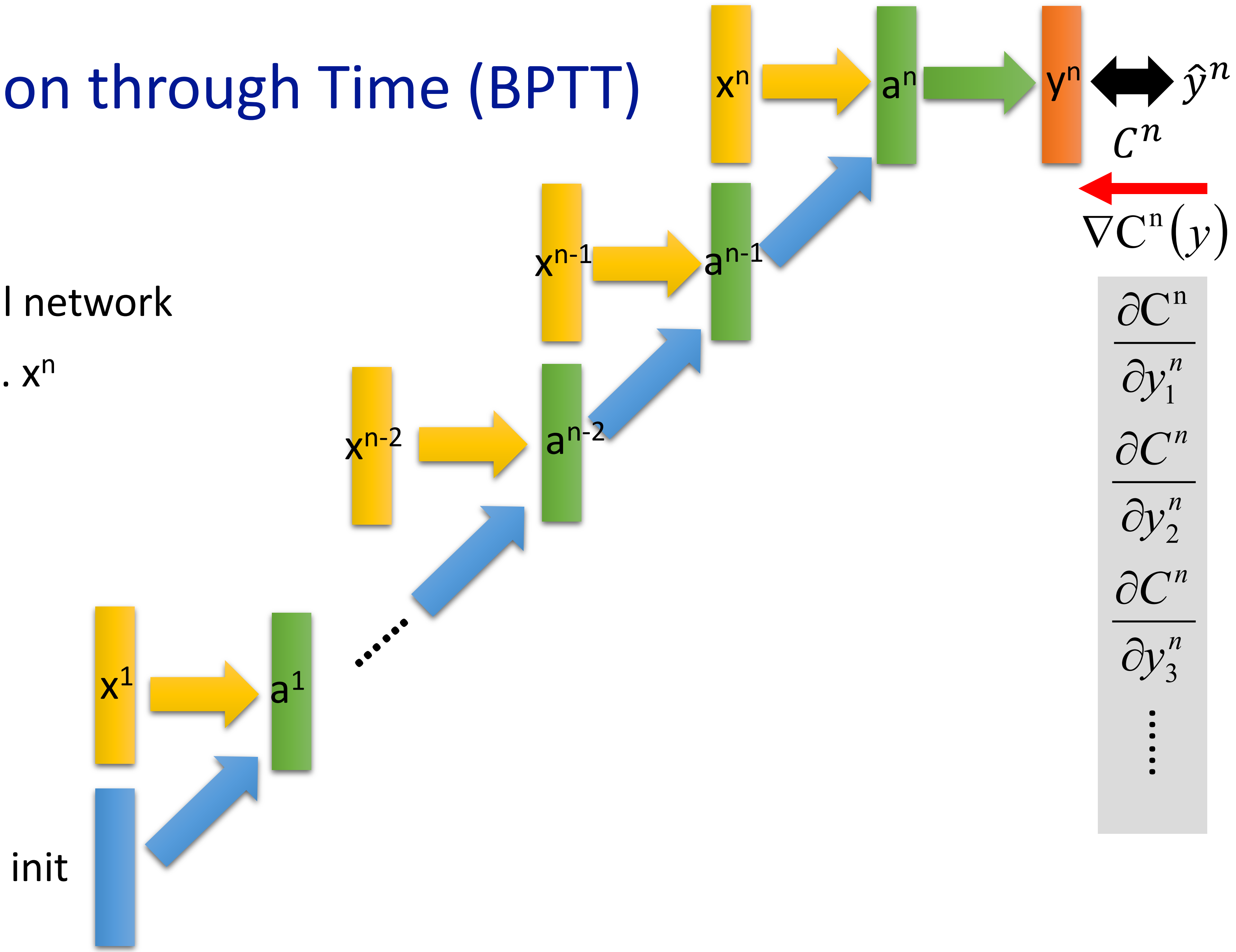
UNFOLD:

A very deep neural network

Input: init, x^1, x^2, … x^n

output: y^n

target: $\hat{y}^n$



Backpropagation through Time

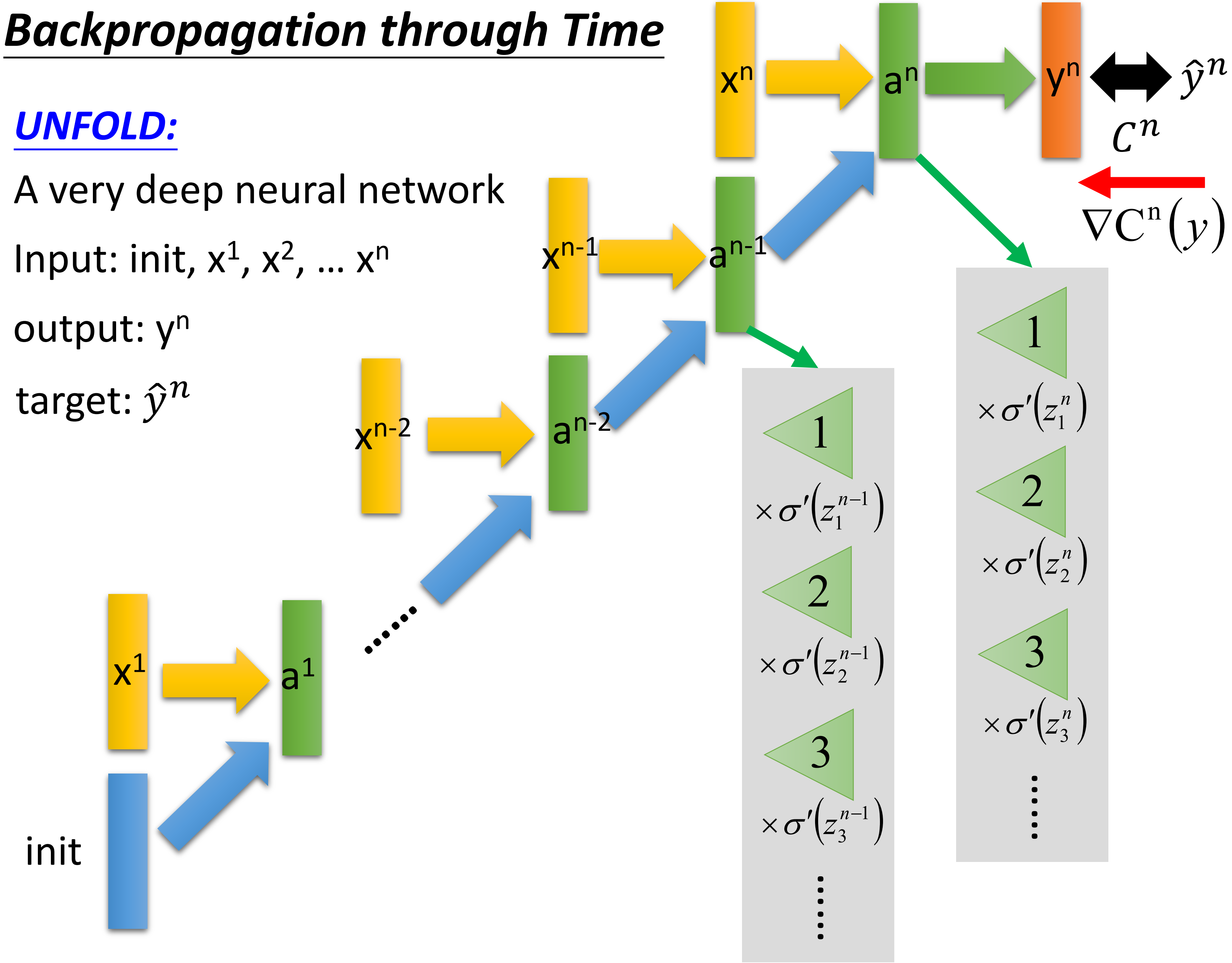
UNFOLD:

A very deep neural network

Input: init, x^1, x^2, … x^n

output: y^n

target: $\hat{y}^n$



Backpropagation through Time

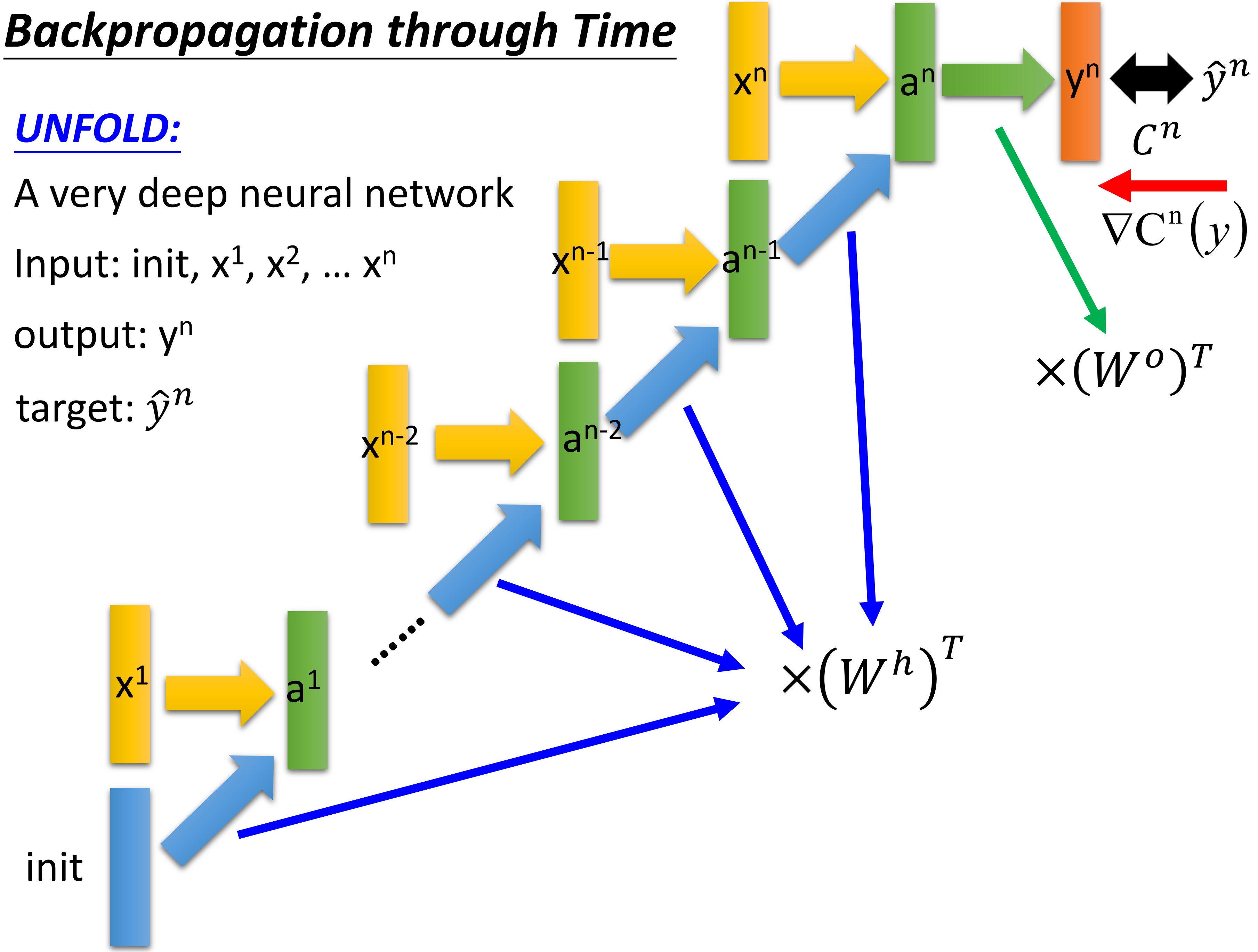
UNFOLD:

A very deep neural network

Input: init, x^1, x^2, … x^n

output: y^n

target: $\hat{y}^n$



Backpropagation through Time

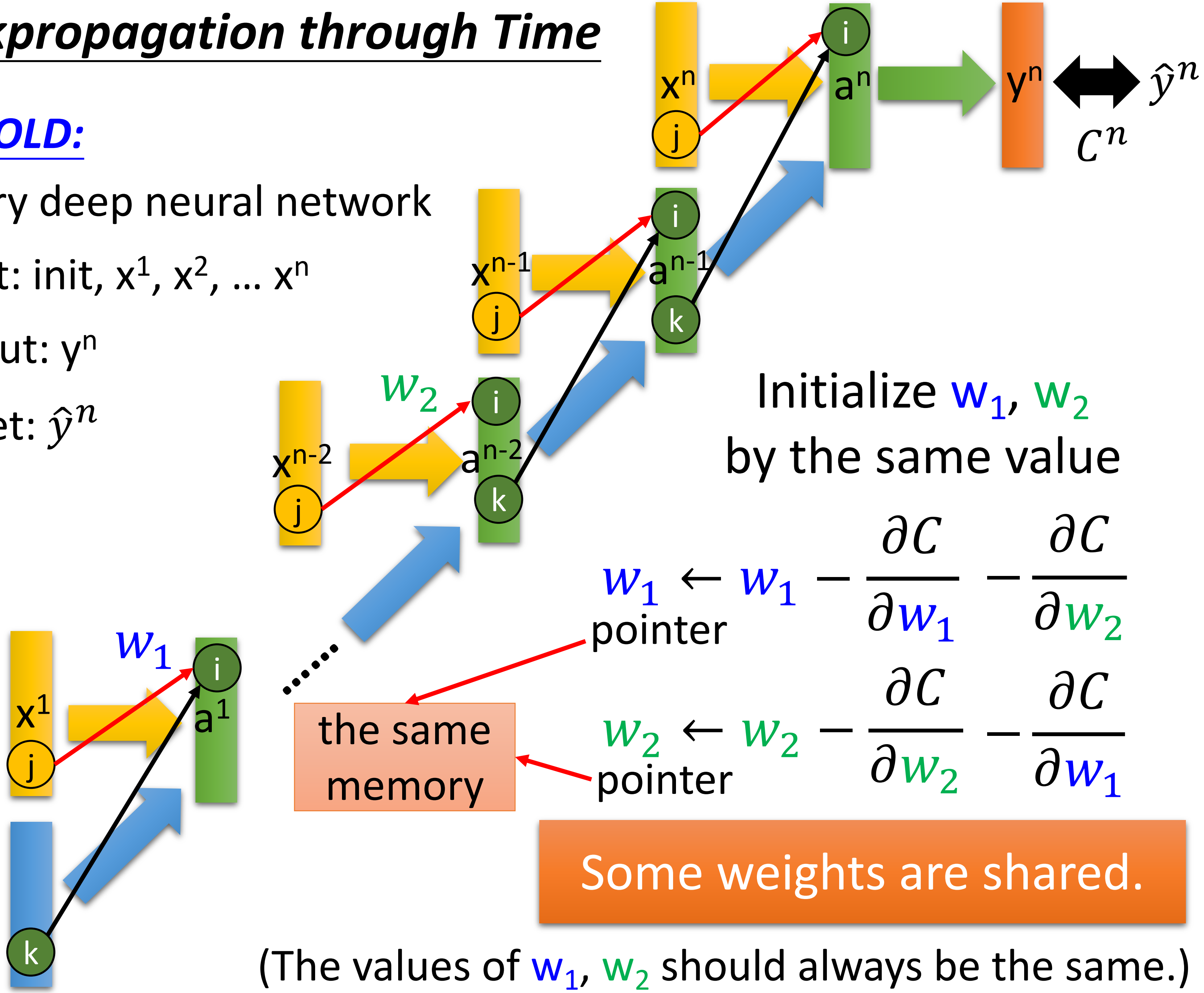
UNFOLD:

A very deep neural network

Input: init, x^1, x^2, … x^n

output: y^n

target: $\hat{y}^n$



Initialize w_1, w_2 by the same value

$$w_1 \leftarrow w_1 - \frac{\partial C}{\partial w_1} - \frac{\partial C}{\partial w_2}$$

pointer

$$w_2 \leftarrow w_2 - \frac{\partial C}{\partial w_2} - \frac{\partial C}{\partial w_1}$$

pointer

Some weights are shared.

(The values of w_1, w_2 should always be the same.)

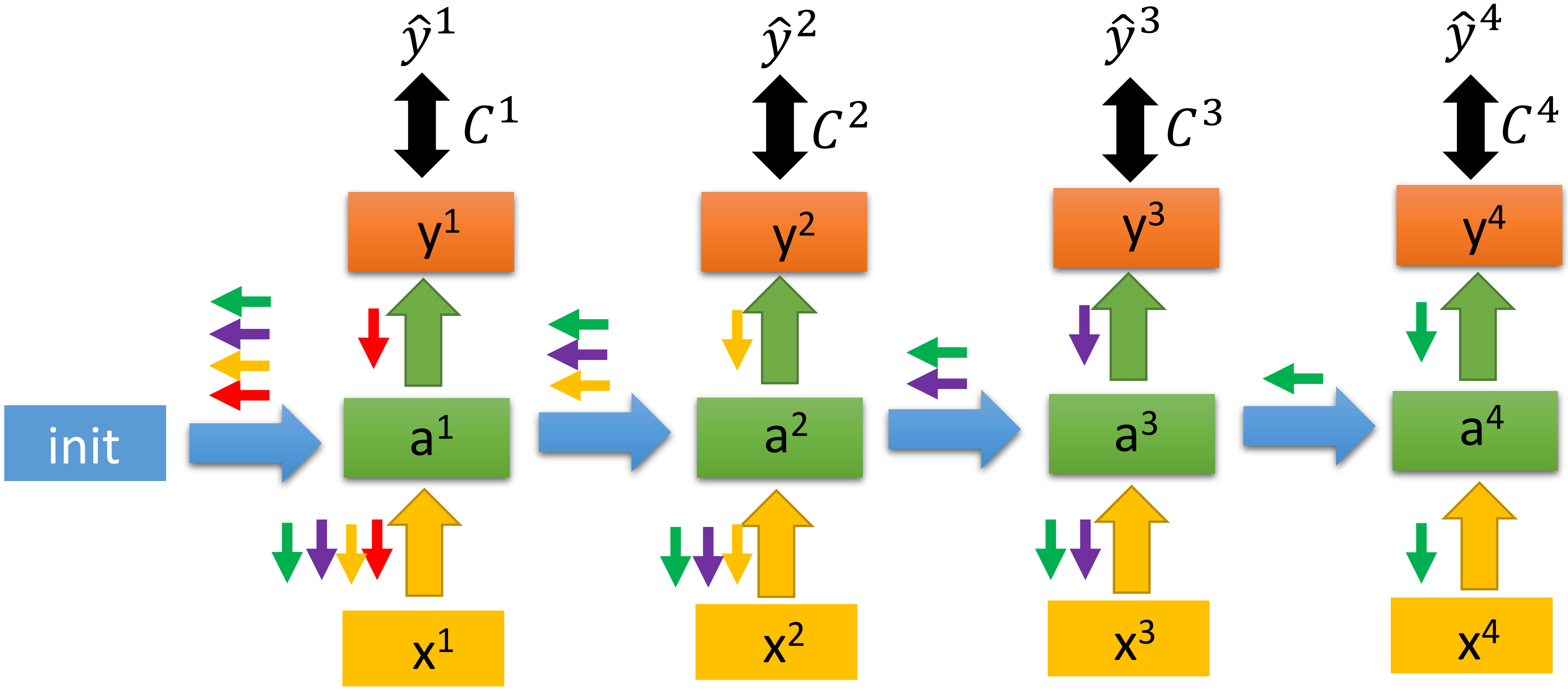
BPTT

Forward Pass: Compute a^1, a^2, a^3, a^4 ……

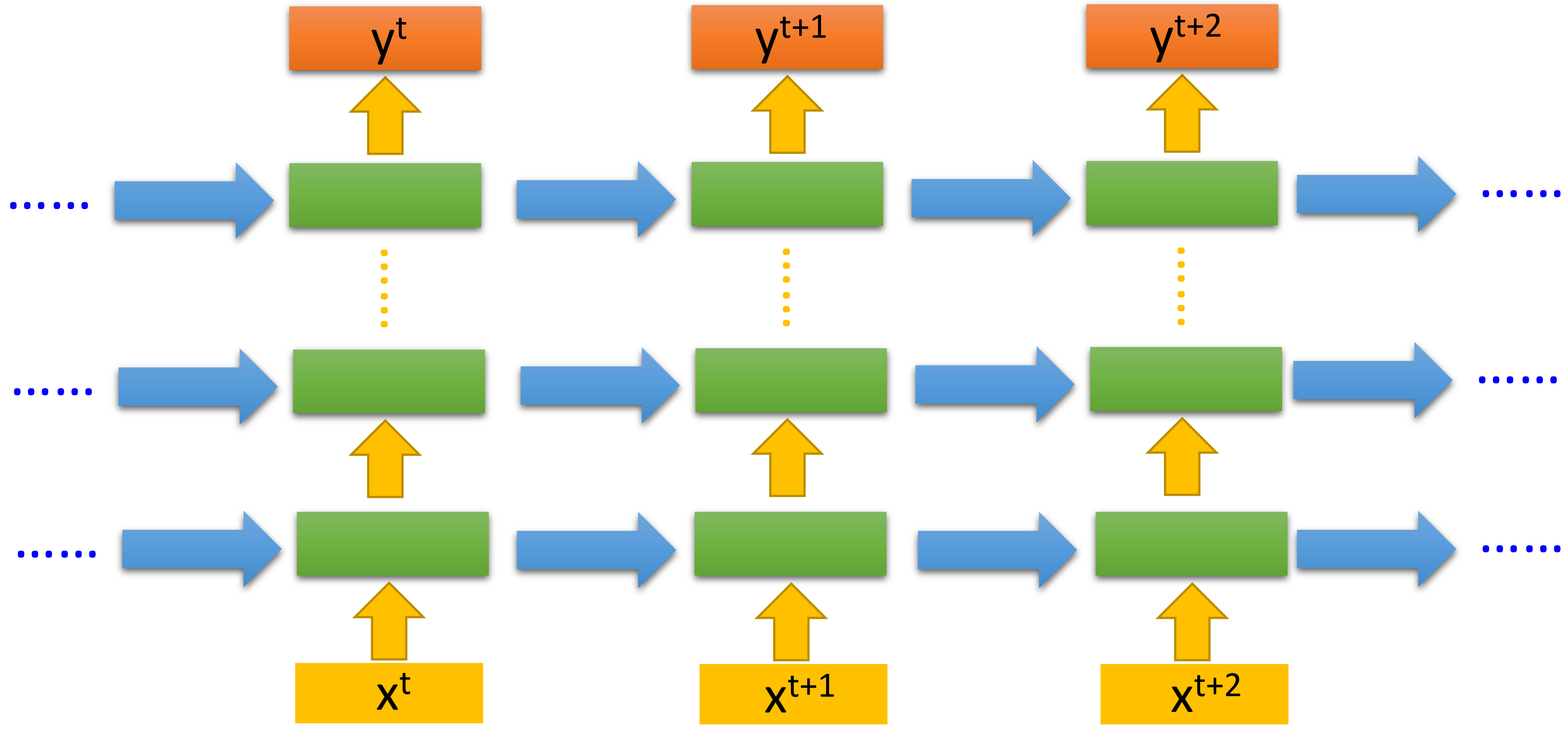
Backward Pass:

For C^4 For C^3

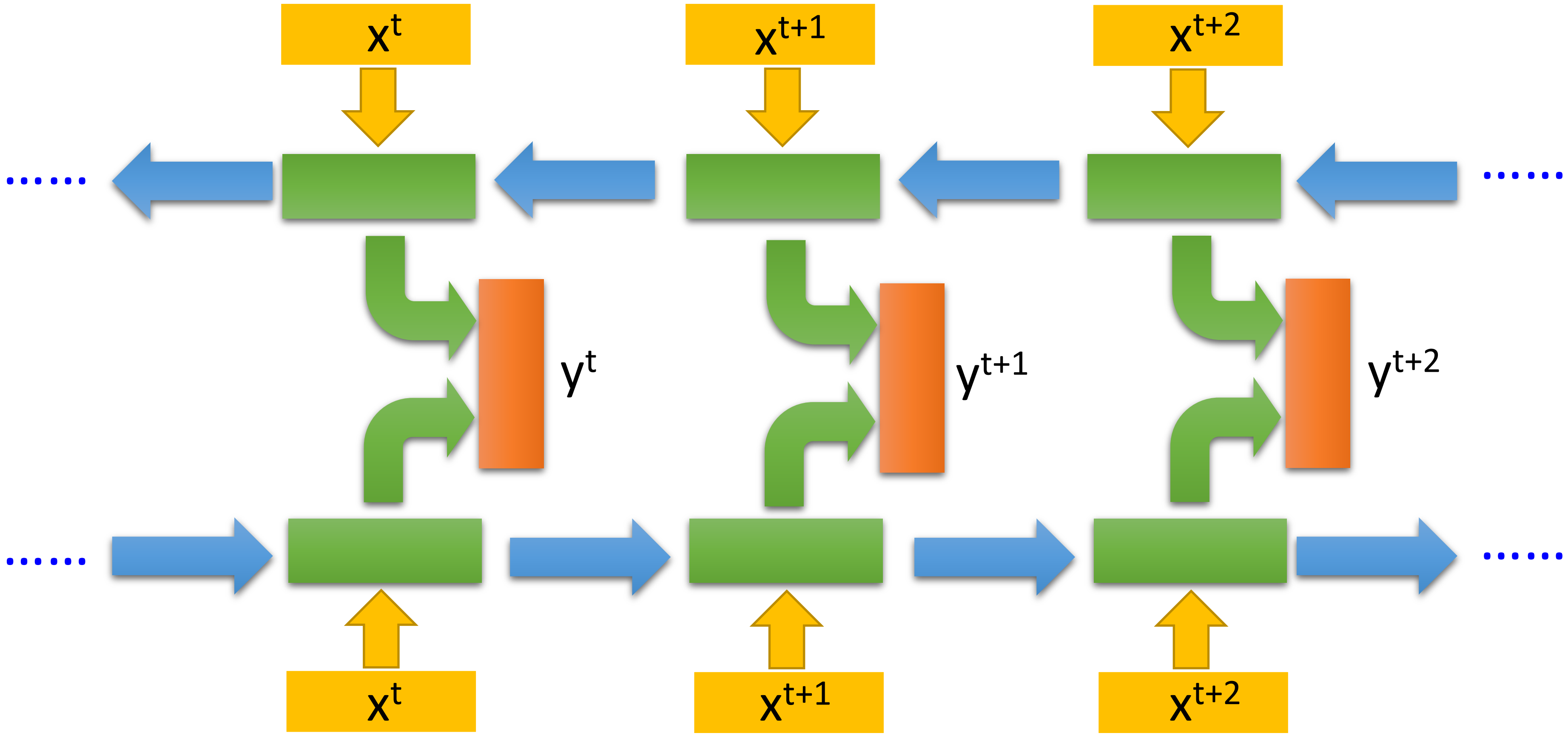
For C^2 For C^1



Deep RNN



Extension: Bidirectional RNN



Bi-directional RNNs

- I was trying really hard to get a hold of ______. **Louise**, finally answered when I was about to give up.

her
him
them

- Louise → him

RNN Training Issue

- The gradient is a product of Jacobian matrices, each associated with a step in the forward computation
- Multiply the same matrix at each time step during backprop

$$\delta^l = \sigma'(z^l) \odot (W^{l+1})^T \delta^{l+1}$$

The gradient becomes very small or very large quickly
➔ vanishing or exploding gradient

Solutions for Vanishing or Exploding Gradients

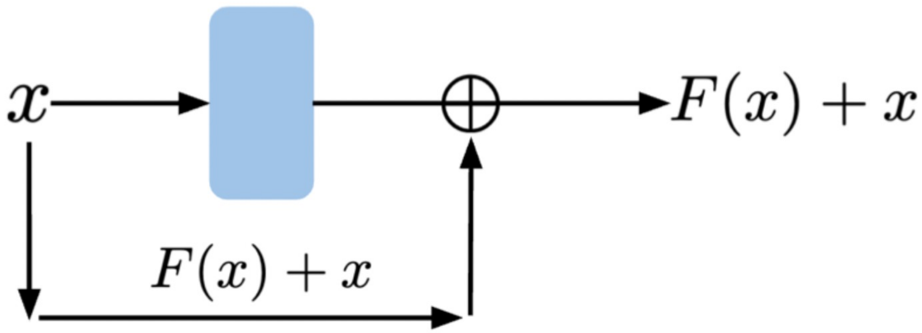
- Identify RNN with ReLU activation

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

-1 → 0

- Gradient clipping 32 → 25

- Skip connections



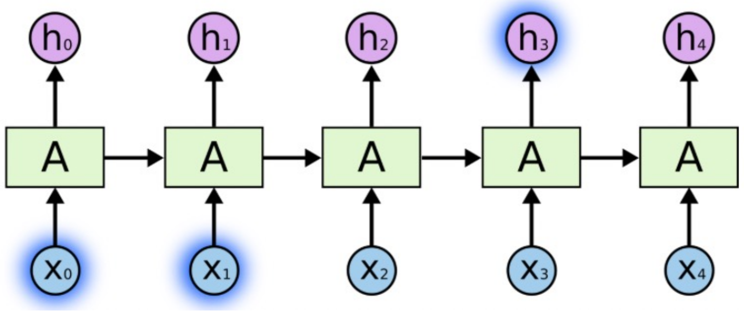
Summary

- RNNs: Advantages
- Captures dependencies within a short range
- Takes up less RAM than other n-gram models

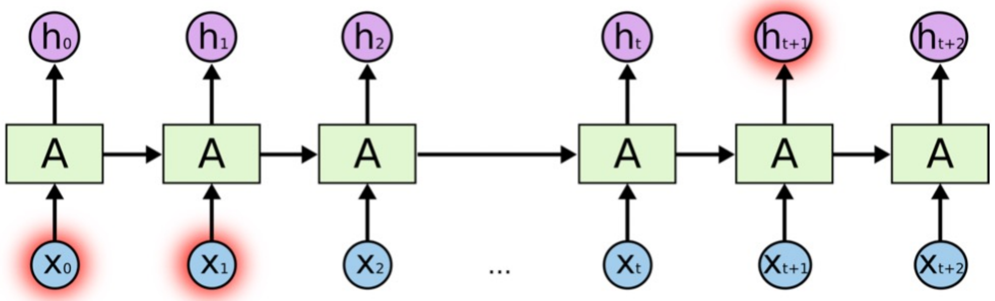
- RNNs: Disadvantages
- Struggles to capture long term dependencies
- Prone to vanishing or exploding gradients

Gating Mechanism

- RNN models temporal sequence information
  - can handle "long-term dependencies" in theory



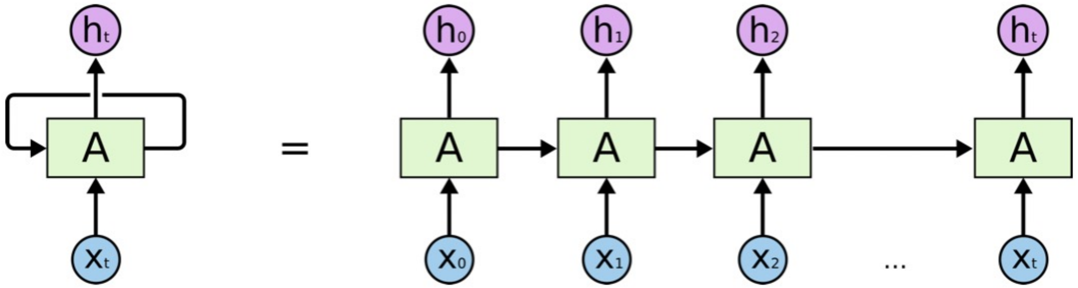
Issue: RNN cannot handle "long-term dependencies" due to vanishing gradient
➔ gating directly encodes long-distance information



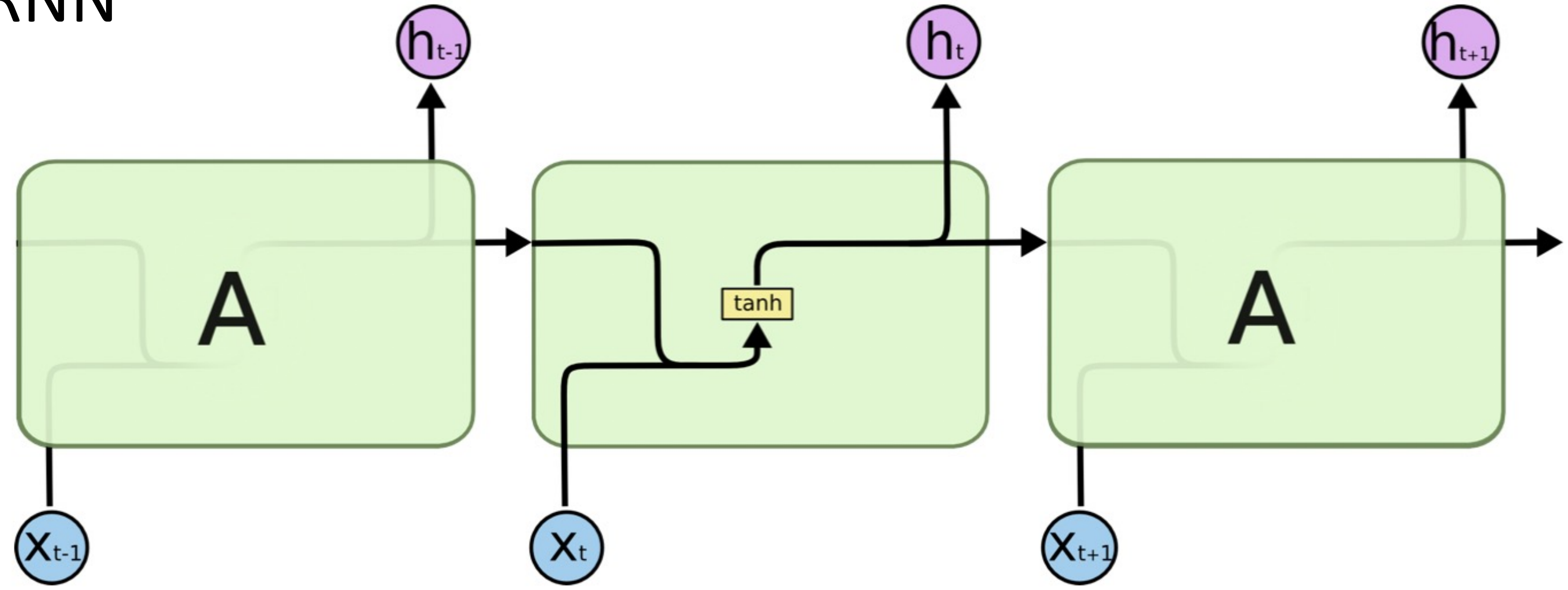
Source of image: http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Long Short-Term Memory (LSTM)

- LSTMs are explicitly designed to avoid the long-term dependency problem

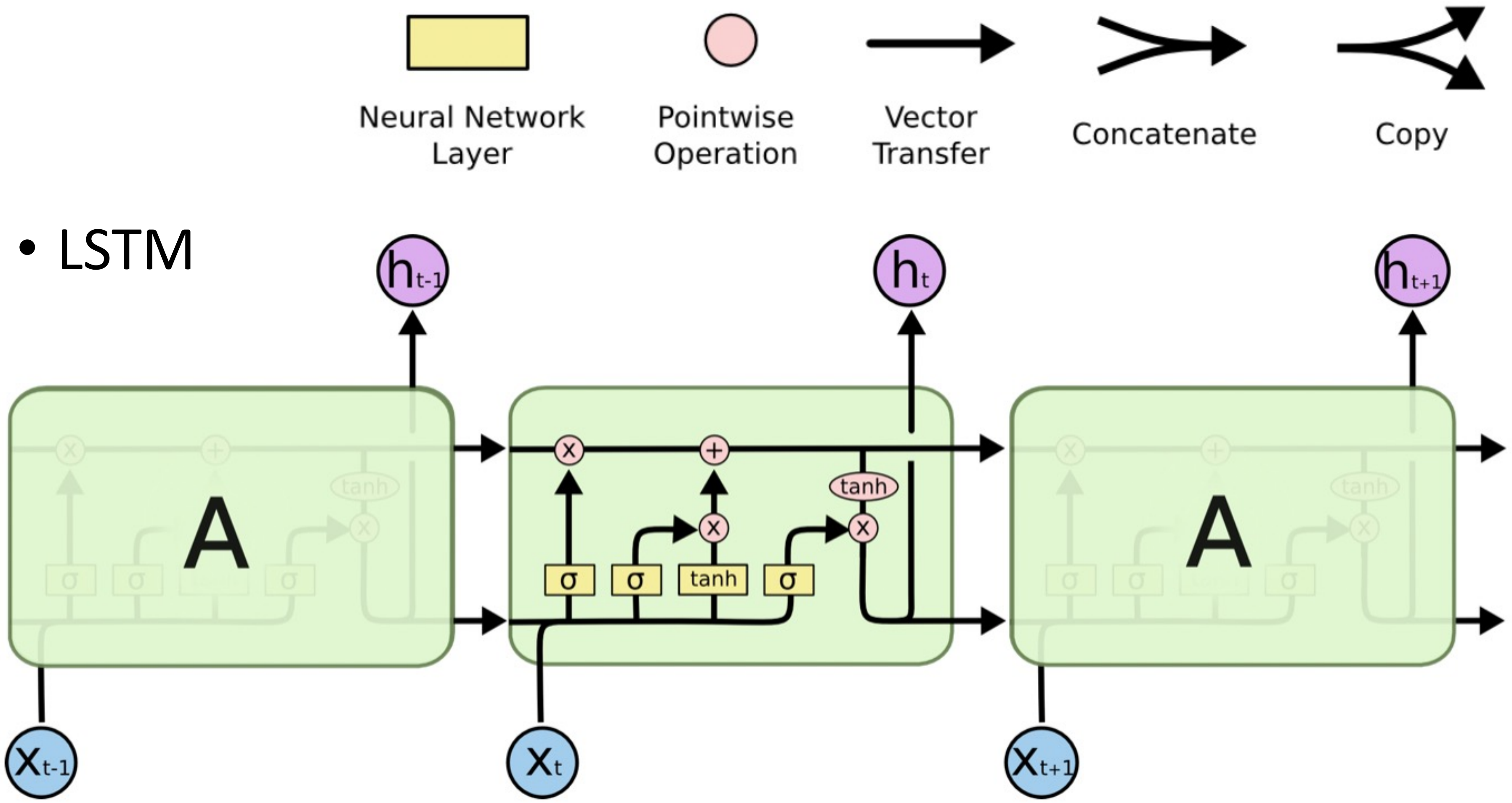


- Vanilla RNN



Source of image: http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Long Short-Term Memory (LSTM)

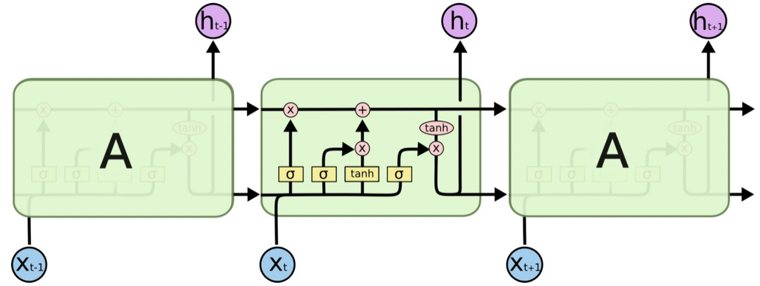


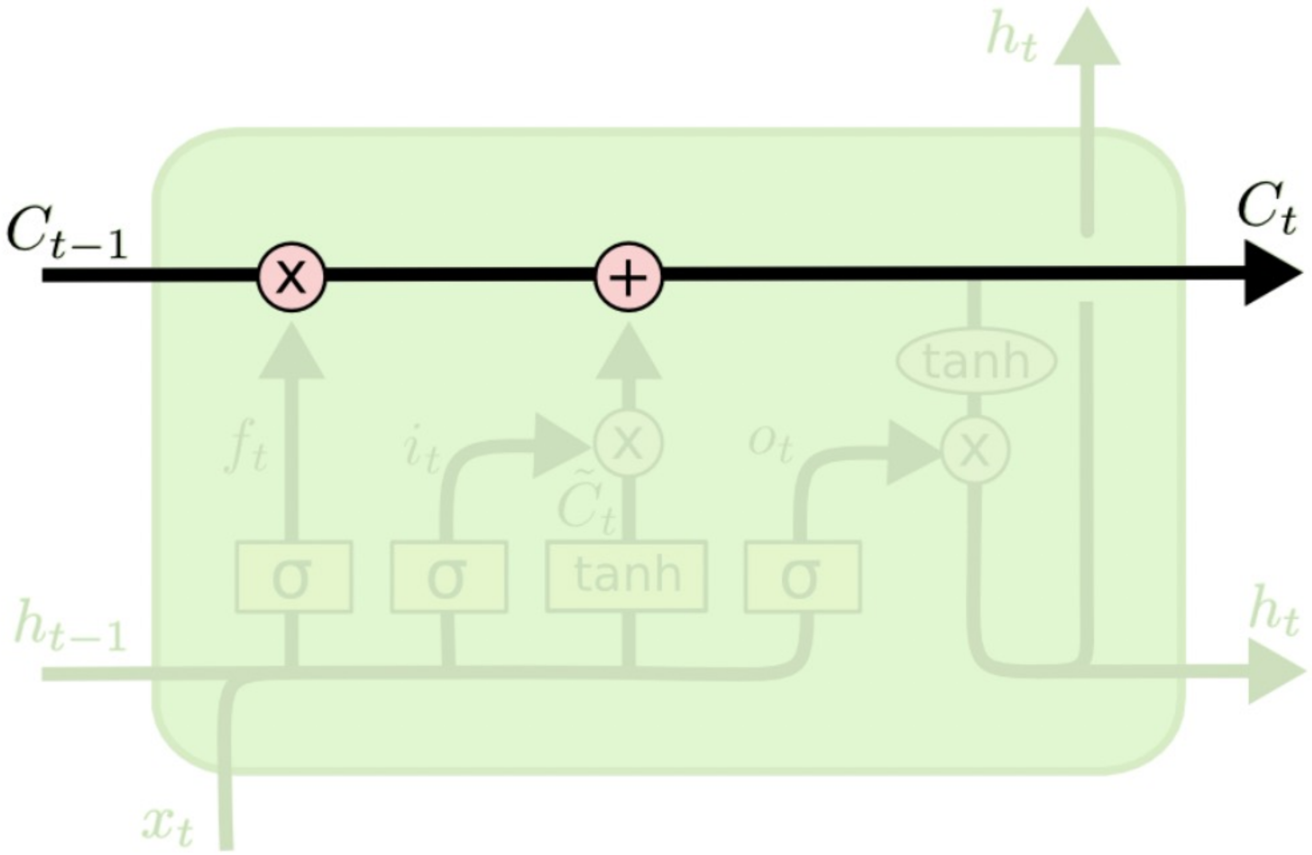
- LSTM

Sigmoid output between 0 and 1
0 → close
1 → open

Source of image: http://colah.github.io/posts/2015-08-Understanding-LSTMs/

LSTM





runs straight down the chain with minor linear interactions

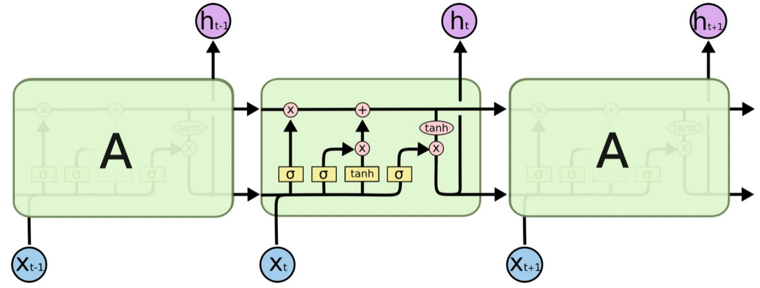
➔ easy for information to flow along it unchanged

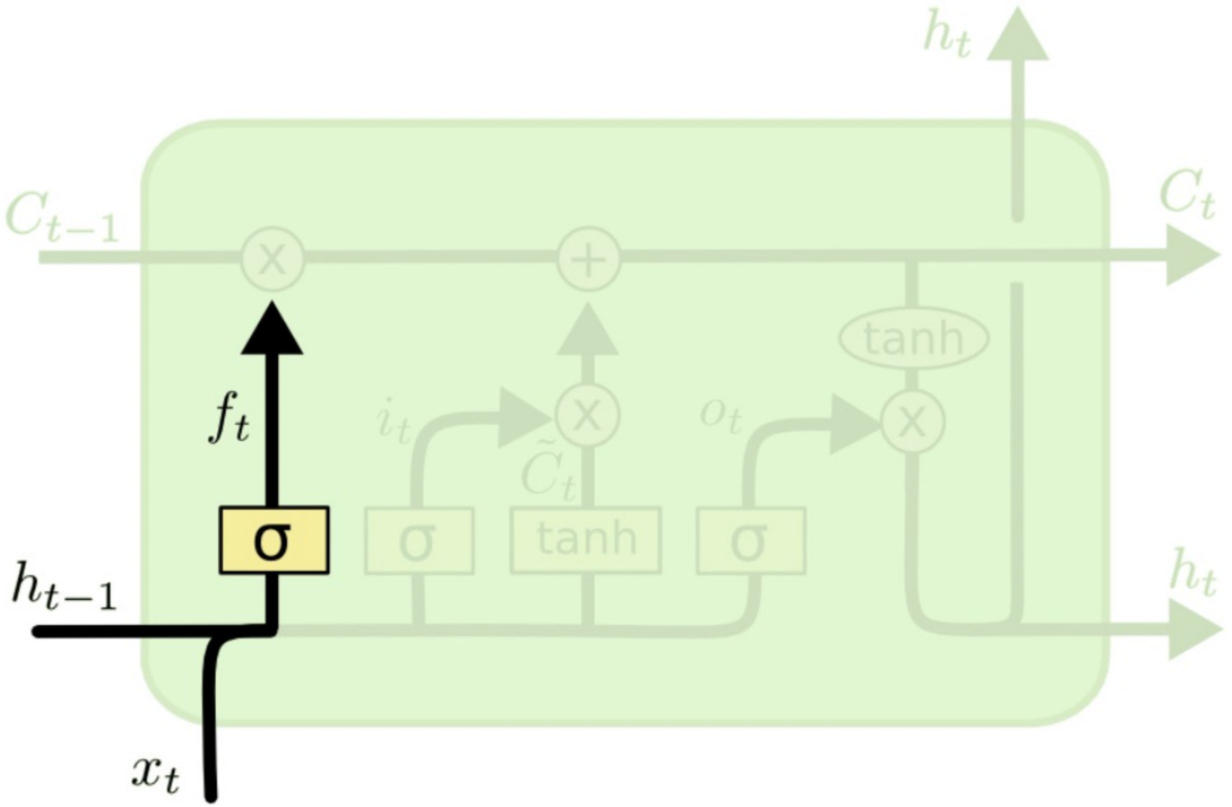
Gates are a way to optionally let information through

➔ composed of a sigmoid and a pointwise multiplication operation

Source of image: http://colah.github.io/posts/2015-08-Understanding-LSTMs/

LSTM





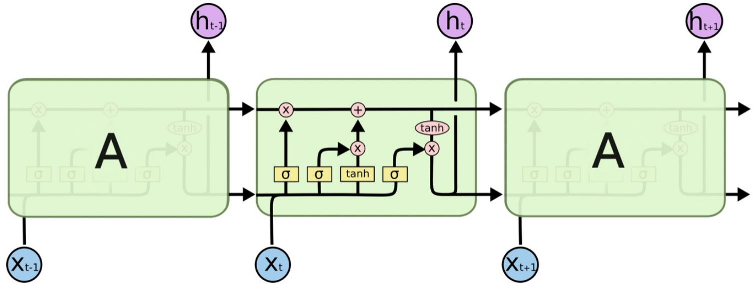
- Forget gate (a sigmoid layer): tells you how much information to forget at any time point.

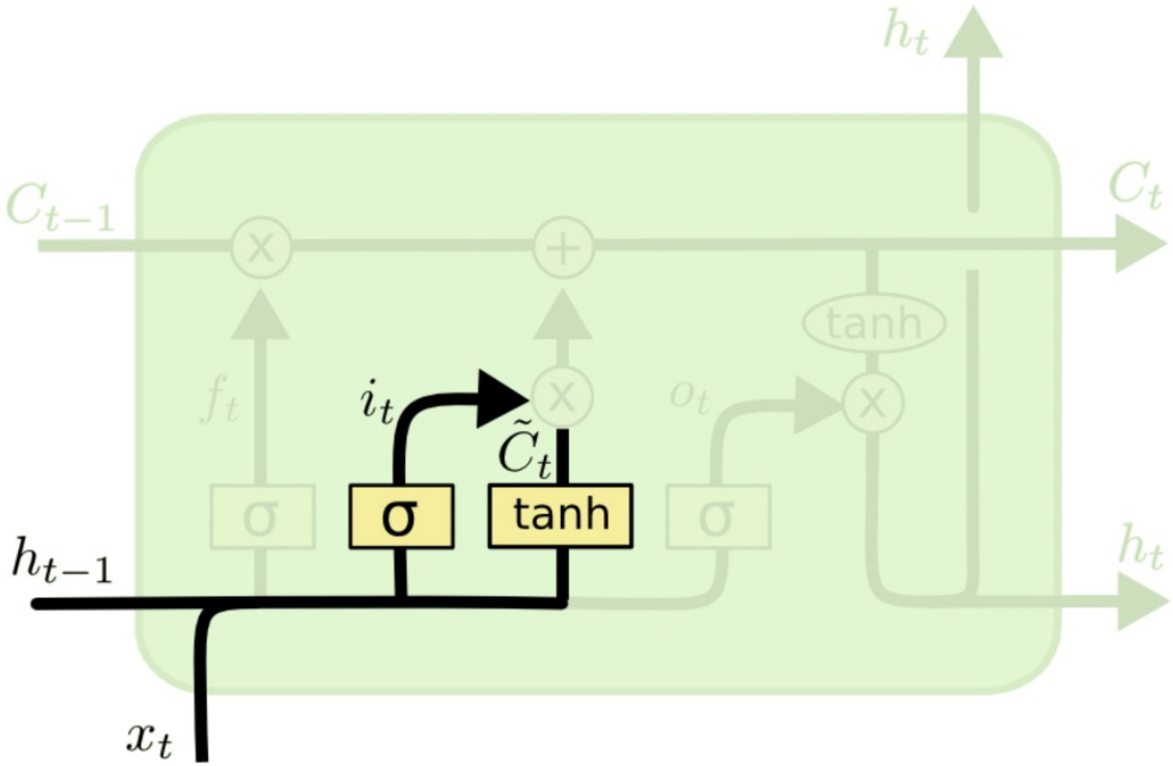
$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

- 1: “completely keep this”
- 0: “completely get rid of this”

Source of image: http://colah.github.io/posts/2015-08-Understanding-LSTMs/

LSTM





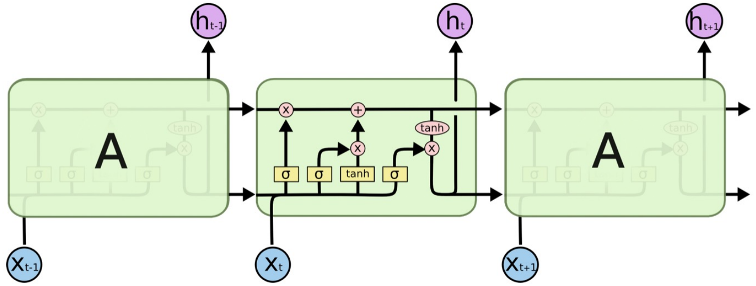
- Input gate (a sigmoid layer) : tells you how much information to input at any time point.

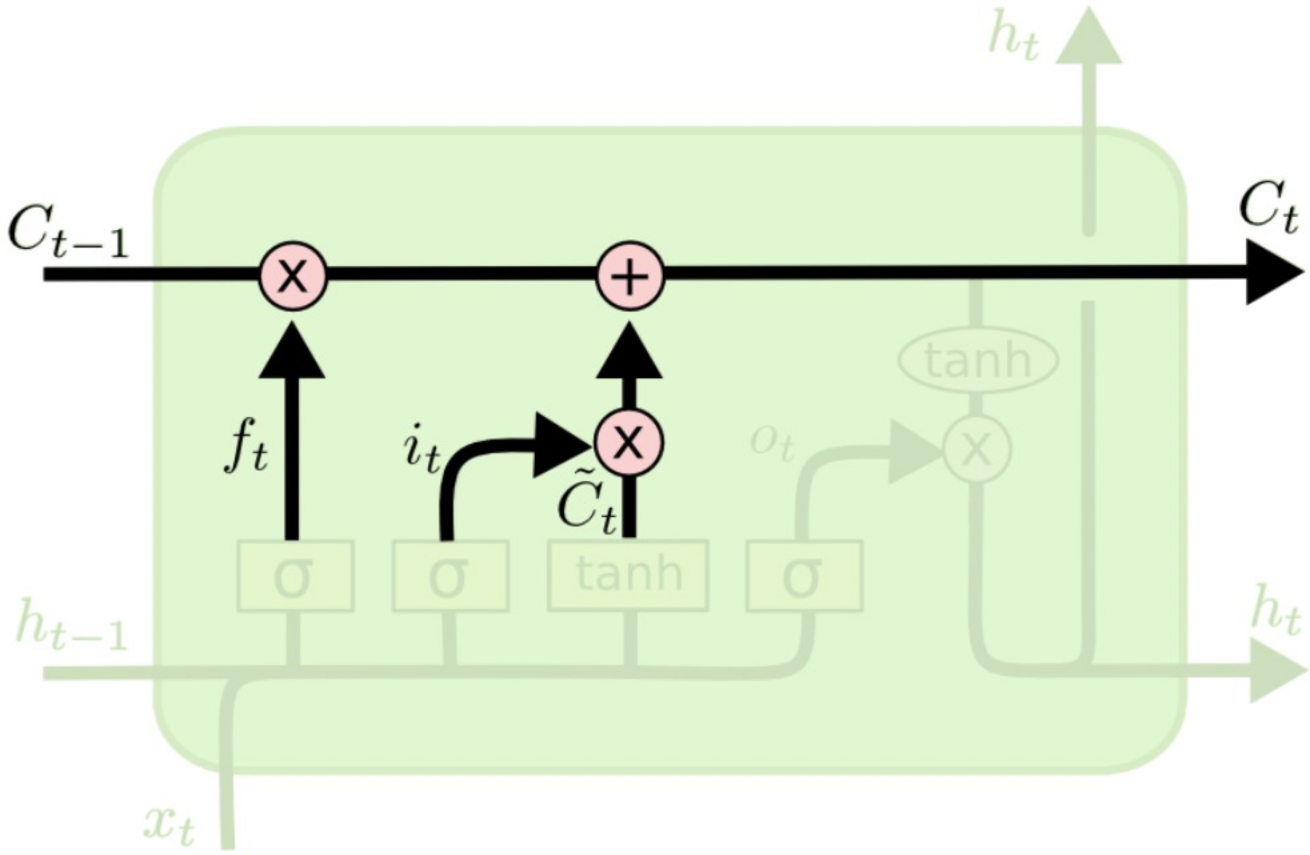
$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \ + \ b_i\right)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \ + \ b_C)$$

Source of image: http://colah.github.io/posts/2015-08-Understanding-LSTMs/

LSTM





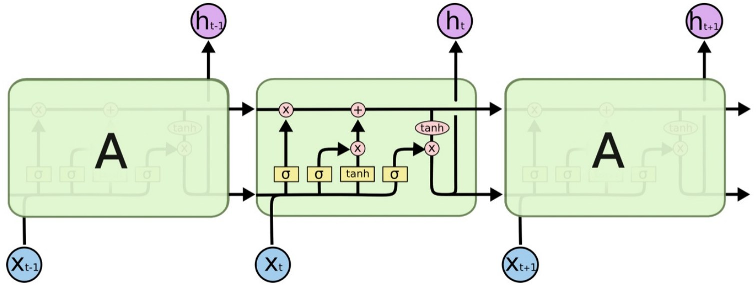
- cell state update: forgets the things we decided to forget earlier and add the new candidate values, scaled by how much we decided to update each state value

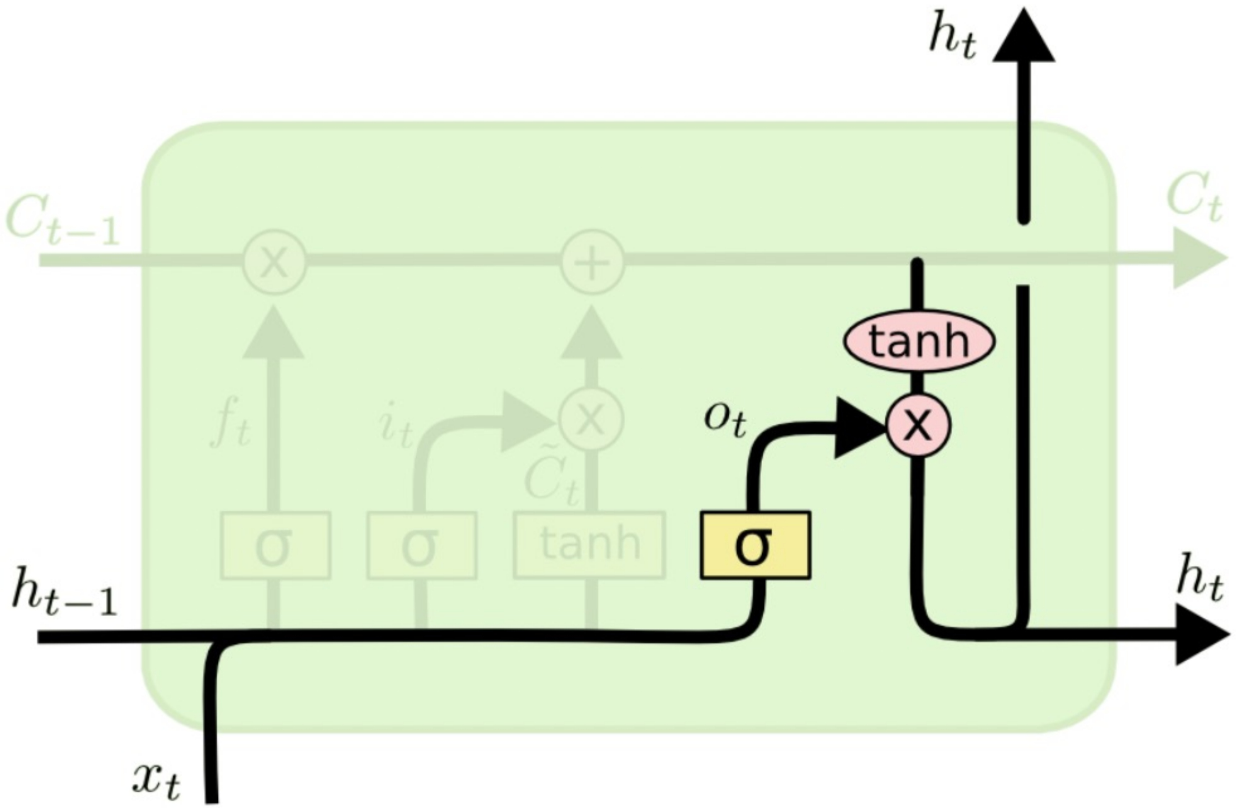
$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

- f_t : decides which to forget
- i_t: decide which to update

Source of image: http://colah.github.io/posts/2015-08-Understanding-LSTMs/

LSTM





- Output gate (a sigmoid layer) : tells you how much information to pass over at any time point.

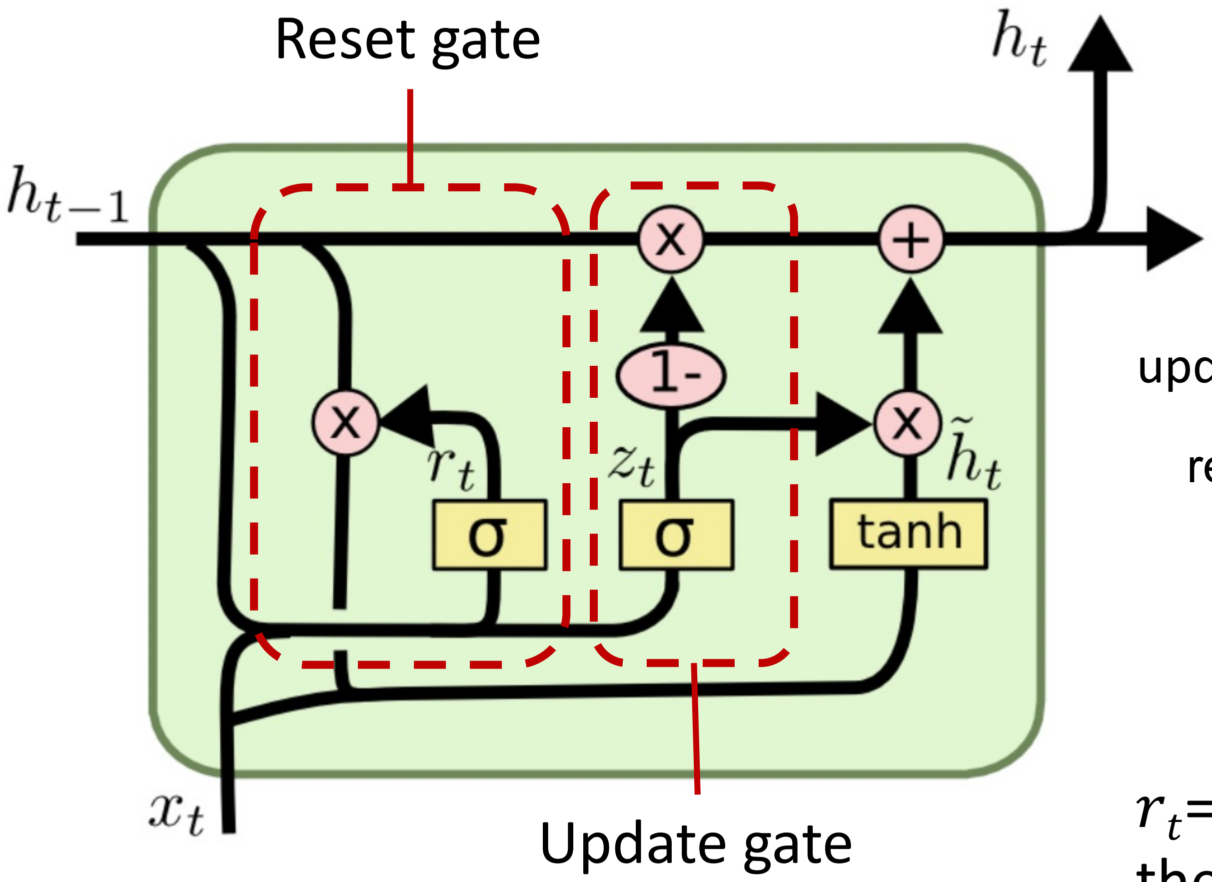
$$o_t = \sigma\left(W_o \left[h_{t-1}, x_t\right] + b_o\right)$$

$$h_t = o_t * \tanh\left(C_t\right)$$

Source of image: http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Gated Recurrent Unit (GRU)

GRU is simpler and has less parameters than LSTM



- combine the forget and input gates into a single "update gate"; merge the cell state and hidden state

update gate $z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$

reset gate $r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$

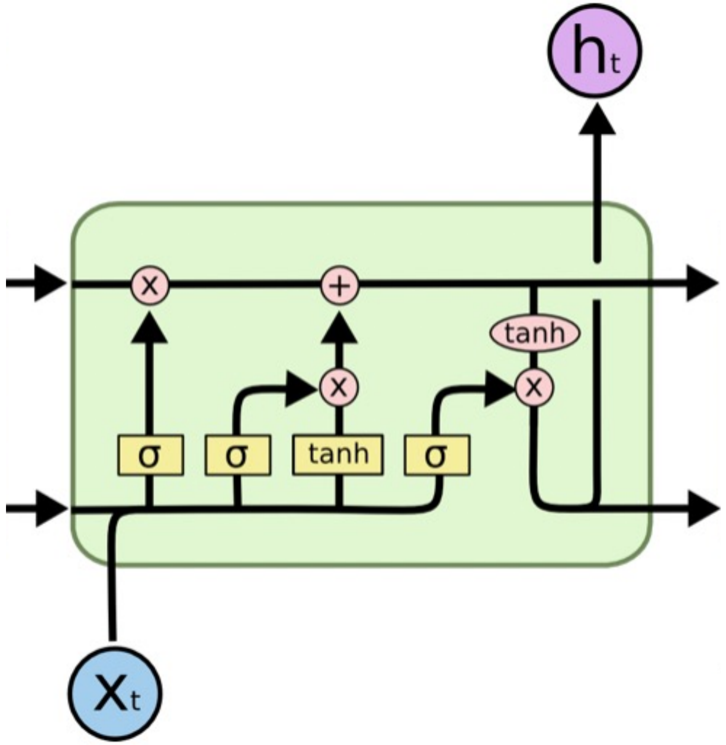
$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

r_t=0: ignore previous memory and only stores the new word information

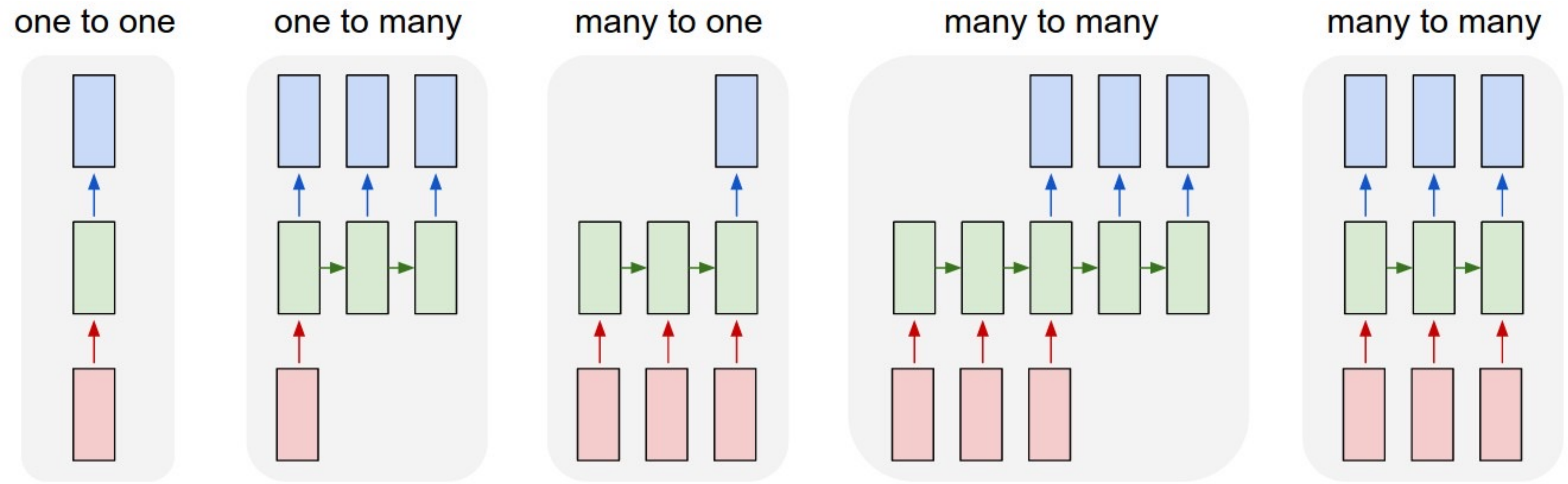
Source of image: http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Concluding Remarks



Long Short-Term Memory (LSTM) :

- Input gate: tells you how much information to input at any time point.
- Forget gate: tells you how much information to forget at any time point.
- Output gate: tells you how much information to pass over at any time point.



Each rectangle is a vector and arrows represent functions (e.g. matrix multiply). Input vectors are in red, output vectors are in blue and green vectors hold the RNN's state

Source of image: http://karpathy.github.io/2015/05/21/rnn-effectiveness/

Applications of LSTMs

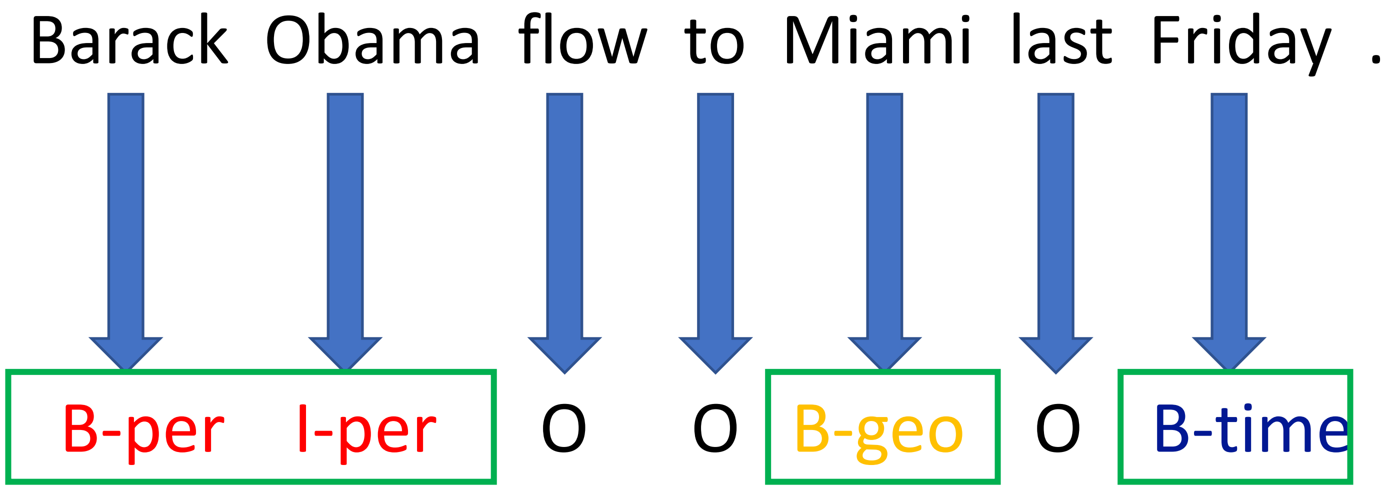
- Next-character prediction
- Chatbots
- Image captioning
- Speech recognition
- Music composition

Applications In NLP: Sequence Labeling

- Part-of-speech Tagging (POS)
- Named Entity Recognition (NER)
- Text Chunking
- Others: dependency parsing, semantic role labeling, answer selection, text error detection, document summarization, constituent parsing, sub-event detection, emotion detection in dialogue

Named Entity Recognition

- Locates and extracts predefined entities from test
- Places, organization, names, times, and dates
- Labeling: BIO Encoding



IO, BIO, and BIOES taggings

Words	IO Label	BIO Label	BIOES Label
Jane	I-PER	B-PER	B-PER
Villanueva	I-PER	I-PER	E-PER
of	O	O	O
United	I-ORG	B-ORG	B-ORG
Airlines	I-ORG	I-ORG	I-ORG
Holding	I-ORG	I-ORG	E-ORG
discussed	O	O	O
the	O	O	O
Chicago	I-LOC	B-LOC	S-LOC
route	O	O	O
.	O	O	O

Reference

- Lecture Slides from Applied Deep Learning by Prof. Yun-Nung Chen
- https://www.csie.ntu.edu.tw/~miulab/s110-adl/doc/220221_RNN.pdf
- https://www.csie.ntu.edu.tw/~miulab/s110-adl/doc/220314_Attention.pdf

- Lecture Slides from Machine Learning And Having It Deep And Structured by Prof. Hung-Yi Lee
- https://speech.ee.ntu.edu.tw/~tlkagk/courses/MLDS_2015_2/Lecture/RNN%20training%20(v6).pdf

- Natural Language Processing with Sequence Models
- https://www.coursera.org/learn/sequence-models-in-nlp

- Natural Language Processing with Attention Models
- https://www.coursera.org/learn/attention-models-in-nlp